Making the Transition from *R-code* to *Arc*

Sanford Weisberg
Supported by the National Science Foundation
Division of Undergraduate Education Grants 93-54678 and 96-52887.

May 17, 2000

Arc is the revision of program *R-code* that is described in R. D. Cook and S. Weisberg (1994) *An Introduction to Regression Graphics*, published by John Wiley & Sons, called *IRG* in the following. *Arc* includes a number of evolutionary changes in the *R-code* that we think are more intuitive than the original, and will make the program easier, and more enjoyable, to use. As a result, some of the descriptions of *R-code* in *IRG* are incorrect. This document is designed to help you make the transition from *R-code* to *Arc*.

A more complete description of *Arc* is contained in the book R. D. Cook and S. Weisberg (1999), *Applied Regression Including Computing and Graphics*, published by John Wiley & Sons. Additional information is also available at the Web address www.stat.umn.edu/arc.

1 Getting the Program

Arc is distributed on the internet only. The current web address is given above. Complete instructions for down-loading are given on the website, or on the installer files themselves.

2 The Arc Directory

The files in *Arc* are a little different than in the original *R-code* files. First, there is a new version of *Xlisp-Stat*, version 3.52 or later. There is no longer a folder called `R-code` that contains source code. *Arc* is included in a *workspace* `xlisp.wks`. The sample data files have been moved to a folder called `Data` (the data files for *IRG* are available from the website and require a separate down-load). There are a few additional files as well. Also included is a directory called `Extras`. This directory is initially empty, but any file you put in this directory whose name ends in `.lsp` will be loaded automatically whenever the program is started. For example, from time to time, updates to *Arc* are posted on the website; you should down-load the `updates.lsp` file and put it in your `Extras` directory, following the directions on the web.

3 Starting Arc

With the Mac OS, *Arc* is started by double-clicking on the icon for either Xlisp-Stat or for the file `xlisp.wks`. With Windows 95 or newer, start *Arc* using the Start button, by first selecting Programs, then Arc, and then Arc again. For the Unix/Linux version, *Arc* will generally be started by simply typing `arc` in a text window.

4 Differences between R-code and Arc

The version of *R-code* described in *IRG* took a *model* as its basic object. Data would be read from a file, and immediately a regression based on those data would be computed. Menus were then created that corresponded to the model that was fit.

In *Arc*, the basic object is a *data set*. When a data file is read, the data set is created. From the data set one can modify the data, create models and draw graphs. This approach is much more general, and we think corresponds to the way people really think about data.

4.1 The Arc Menu

When you start *Arc* on any system, before you load any data files, you will get a menu called Arc. This menu has the following items:

Load Select this item to load: (1) a previously prepared *Arc* data file, such as all the files included with *Arc* distribution, or any file created using *Arc*; (2) any file of *Xlisp-Stat* code; or (3) a "plain" datafile.

Dribble... Save output to a file; see *IRG*, p. 12–13.

Calculate probability... Calculate probabilities for Normal, t, Chi-squared, or F distributions. Input values of the relevant statistic and degrees of freedom and the upper, lower, or two tailed probability is returned. A help button on the dialog provides further documentation.

Calculate quantile... Calculate a quantile for a Normal, t, Chi-squared, or F distribution. Input values of the relevant significance level and degrees of freedom and the one or two tailed quantile is returned. A help button on the dialog provides further documentation.

About Arc Copyright information.

Help... In the resulting dialog, type the name of a function to get help on its usage, if help is available (as an example, select this item and type `recode` in the dialog), or to find all functions that include that text string in their name.

Settings... Change and save settings of various features used in *Arc*. In the dialog, select a setting you wish to change, and choose "Update selection" from the menu. If you change a number of settings, you can return them to their default values by selecting "Change all settings to defaults". Any changes you make will be for the current session only unless you save the changes by selecting "Save settings file". This will create a file called `settings.lsp` that will be read every time you start *Arc*.

Quit Quit *Arc* and *Xlisp-Stat*.

The Arc menu will also list the names of the the data sets you have read into the program. You can then switch between data sets by selecting the name of the data set you want from the Arc menu.

5 Data Files

Data files prepared for *R-code* can be directly read into *Arc*. However, several other file formats are available as well.

5.1 Plain Data

The simplest format of a data file is a *plain data file*, which is a text file with *cases* given by rows and *variables* given by columns. The file can have any legal name. To read this file, simply select Load from the Arc menu (or use the Load item on from the File menu on the Macintosh or Windows). You will be prompted via dialog boxes to name the data set, to type a description of the data, and, if the file has no row that contains variable names, to name variables and give a description of them. You will be able to save the data set as a formatted *Arc* data file using the item "Save data set as..." in the data set menu (the exact name of this menu is the same as the name you give to the data set you are using).

Utilities are available on the Arc website for reading files in this format when using Windows 95 or newer directly from Microsoft Excel into *Arc*, and for transferring data from SAS to *Arc*.

5.2 Getting into Trouble with Plain Data Files

When *Arc* reads a plain data file, it determines the number of columns of data by counting the number of items on the first row of the file. If your data file has many variables, the program that created the file (like an editor or spreadsheet program) may have split data into more than one line, and *Arc* will get the wrong count. You will then get the following error message:

```
Error: list not divisible by this length
```

You can get *Arc* to read the file correctly by typing a command in the text window. If your data file, say `fred.lsp`, has 17 variables, type:

```
> (arc :data (read-data-columns "fred.lsp" 17))
```

If you want the program to find the data file for you, you can type

```
> (arc :data (read-data-columns (open-file-dialog) 17))
```

If this doesn't work, here are a few standard *Xlisp-Stat* functions that can help you. Type

```
> (setf d (read-data-file "fred.lsp"))
```

to set `d` equal to a list that contains the contents of the data file, with the contents of the first row, then the contents of the second row, and so on. Type

```
> (length d)
```

to tell you how many data elements are in the data file in total. Type

```
> (select d (* 17 (iseq 4)))
```

to print a few elements of `d`, namely elements 0, 17, 34 and 51. If the data file had 17 columns, this would give the first four items in the first column of the file. You can often find problems using these functions, and then use a text editor to correct the data file.

5.3 Formatted Data File

Files created using the "Save data set as..." item in the data set menu will have a simple, human-readable format. You can also create files using this format. Table 1 shows a formatted data file.

The first few lines of the file contain information *Arc* uses to assign names, labels, and so on. Statements are of two types: assignment statements, like `data set = hald`, which assigns `hald` to be the name of the data set, and constructions that start with a `begin` and continue until an `end` is reached.

The assignment `columns = 5` specifies that the file has five columns. Specifying the number of columns is required only if some of the columns in the data are not to be used. The assignment `missing = ?` sets the missing value place holder; missing values are discussed in Section 5.5. If the `delete-missing` assignment is equal to `t`, then all cases from the data that have any missing values will be removed from the data set. If `delete-missing` is `nil`, then all the data are used and missing cases are deleted as necessary in the computations. The first option guarantees that all graphs/models are based on exactly the same observations, while the latter option allows each graph/model to use the maximum number of cases possible.

The `begin` ...`end` constructions are used to define blocks of information for *Arc*. The `begin description` construction allows typing in documentation for the data set. The documentation should be fairly short, 10 lines or less, although there is no formal limit. The construction `begin variables` is followed by one line for each column of data to be used that describes the data. These lines are in the format:

$$\text{column number} = \text{variable name} = \text{variable description}$$

The variable name is a short label and the variable description is a longer (about 30 characters or less is good) description of a variable. You should avoid the use

Table 1: A *formatted data file*.

```
data set = hald                 ... The name of the data set is hald.
missing = ?                     ... Sets the missing value place holder.
delete-missing = nil            ... If t, cases with missing data are deleted.
columns = 5                     ... Specifies the number of columns of data.
; a comment to skip             ... Any line beginning with a ; is skipped
begin description               ... The following lines describe the data:
The hald data used in Chapter 1 of An Introduction to Regression
Graphics.
end description                 ... End of the description
begin variables                 ... Begin defining variables; see text for details
Col 0 = X1  = Chemical 1
Col 1 = X2  = Chemical 2 ; another comment
Col 2 = X3  = Chemical 3
Col 3 = X4  = Chemical 4
Col 4 = Y   = Heat of hardness
end variables                   ... End of defined variables
begin lisp                      ... begin lisp instructions
(send hald :make-interaction '("X1" "X2"))
end lisp
begin transformation
X1SQ = X1^2
Z1 = (cut x1 2)
end transformation
begin data                      ... The data follow this statement.
(
   7 26  6 60  78.5
   1 29 15 52  74.3
  11 56  8 20 104.3
  11 31  8 47  87.6
   7 52  6 33  95.9
  11 55  9 22 109.2
   3 71 17  6 102.7
   1 31 22 44  72.5
   2 54 18 22  93.1
  21 47  4 26 115.9
   1 40 23 34  83.8
  11 66  9 12 113.3
  10 68  8 12 109.4
)
```

of mathematical symbols like `+`, `-`, `=` and `/` in the definition of variable names, and you may not use line feeds in the variable description.

Another construction of this type is `begin lisp`. All of the lines between this one and the `end lisp` are sent to the lisp interpreter and executed. As shown in Table 1, an interaction between X1 and X2 will be created and added to the data set. You can put any valid lisp commands here that you like.

Similar to a block of lisp statements is a block of transformations. These are of the form "label = expression", where the expression can use variable names that already exist. The expression can be in usual mathematical notation, or a lisp function. This block starts with `begin transformations` and ends with `end transformations`.

The data follows the line `begin data`. The data should be preceded by a left parenthesis "`(`", and terminated with a right parenthesis "`)`". Although the parentheses are not required, including them speeds up reading the data by a substantial amount.

5.4 Old-style Data Files

Data files that worked with *R-code* version 1 can also be used with *Arc*. You can convert old-style data sets to new-style data sets by reading the file into the *Arc*, and using the "Save datafile" item in the data set menu. New-style data sets can be converted to old-style data sets using the command

```
> (arc-new-to-old "new" "old")
```

where "new" is the name of the data file to be converted, and "old" is the old-style file that will be created.

5.5 Missing Values

Missing values in the datafile are indicated by a `?` unless you change the parameter `*default-missing-indicator*` to some other value using the "Settings..." item in the Arc menu, or you set the `missing =` assignment on the datafile. As described above in Section 5.3, if you have missing data, the program will use as many complete cases as are available in any given calculation. But if you include the command `delete-missing = t` in the datafile (see Table 1), then instead cases with missing data on any variate are removed from the data set for all calculations. With categorical data, the handling of categories with structural zeroes (that is categories which it is impossible to observe, as opposed to

categories which can theoretically be observed but which had an observed count of zero) is described in Section 6.1.

6 The Data Set Menu

When you load a datafile, you are no longer prompted to specify a model; rather you get a menu with the name of the data set in the menu bar; we call this the data set menu. Also, a Graph&Fit menu is added to the menu bar. From the data set menu, you can modify and view data; from the Graph&Fit menu you can obtain graphs and fit models. This is a fundamental, philosophical, change from *R-code* version 1. In addition, only one data set is *active* at a given time. You can change between data sets by selecting the name of the data set in the Arc menu.

6.1 Description of Data

Several items in the data set menu permit printed summaries of the data. These menu items are:

Description Print information about the data set.

Display summaries... Print counts, means, standard deviations, minima, medians, and maxima of the data-lists you specify.

Table data... Print tables of data, or create new data sets.

Display data... Print the data you select.

Display case names Show case names in a window.

The new feature here is the "Table data..." item. If you select this item, you can create tables of the data by adding over *conditioning variables*. For example, suppose two of your variates are eye-color (3 categories) and age (4 categories). If you select eye-color and age as conditioning variables by double-clicking on their names and clicking "OK", then you will get a table displayed on the screen giving the number of observations in each of the $3 \times 4 = 12$ cells. If you also selected another variable (say income) as a variate by clicking **once on its name, and once in the list of variates**, and pushed the mean and standard deviation buttons, the table would consist of the number of observations in each of the cells, along with the mean and standard deviation of income for each of the 12 cells.

If you push the "Make new data set" button, then the tabled data are put into a new data set with its own menu. You can also choose between tabling all data or just *included* cases; for the latter option, if a subset of all cases have been selected (by clicking and dragging the mouse over points in a scatterplot, for example) then only these cases are included. Structural zeroes, as described in Section 5.5, are handled by "counting empty cells as missing". There is an alternative option to count empty cells as zero counts.

The dialog for making tables is relatively complicated, and you should review it carefully before pressing the "OK" button.

6.2 Modifying Data

The data set menu has several items that can be used to modify existing data, or to create new data. Here is a brief discussion of what these items do.

Add a variate... Create a new variate (not in *R-code*). The user must enter an expression that defines the new variable. The expression can be a lisp expression, like `new=(^ (+ a b) 2)`, or it can be in standard mathematical format, like `new=(a+b)^2` (the latter is actually an expression in the computer language C). This feature was not available in *R-code*.

Transform... Create new variates using power or log transformations (modified from *R-code*). Several variates can be transformed at the same time. Logs can be computed to any base.

Make factors... Make factors from existing variates. You can use the buttons in the dialog to choose the way the factor is created. Factors can be defined by dropping the dummy variable for the first level, or with a dummy variable for each level of a factor, as in GLIM, or by using Scheffé (contrast) coding, as in many programs like Macanova and JMP. Factors are named by *Arc* by preappending `{F}`, `{T}`, or `{C}` respectively, to the variable's name.

Make interactions... Make interactions between existing variates and factors.

Set case names... Set the variate to be used to define case names. These are used in graphs to label points. The default is to use the first text variable in the data set.

Delete datalist... Delete an existing datalist (i.e., a variate, factor, or interaction). This needs to be done with care; if you delete `X1`, for example, you can have problems with factors or interactions derived from `X1`.

Rename datalist Change the name of an existing datalist. This can also cause problems if the old name appears in previously defined models.

Save data set as... Save the data set in a file that can be read by *Arc*.

Remove data set Remove the current active data set.

The primary changes from *R-code* are: (1) several transformations, factors, and interactions can be created at the same time; (2) log-transformations can be done to any base you like, with natural logarithms to the base e as the default; (3) the `:add-data` method (see *IRG*, p. 234) can be accessed using the "Add a variate..." menu item. You no longer need to know how to write a lisp expression to create a variable, as you can use standard mathematical formulas, like `new=a*b*(1+exp(c))`, assuming that the variables `a`, `b` and `c` are all defined. Variable names including special symbols like the underline _ or question mark `?`, or a mathematical symbol such as `+`, `-`, `*` or `/` are certain to cause problems. For example, the variable name `plant-ht` will be read incorrectly as `plant` *minus* `ht`. Any variable name including a square bracket like `log[ht]` will not work.

7 Graphics from the Graph&Fit Menu

The Graph&Fit menu has six graphics commands. Here are brief descriptions of these items:

Plot of... The generic plotting method for histograms, and 2D and 3D scatterplots.

Scatterplot-matrix of... Create scatterplot-matrices.

Boxplot of... Create boxplots.

Multi-panel plot of... The user chooses one variable for a fixed axis, and many variables for a changing axis. When the plot is drawn, the user can change between the multiple panels of this plot using a plot control.

Probability plot of… Create probability plots.

Set marks When creating a plot, there will be an item called "Mark by…". If you fill in this item, then the variable you select will be used to mark (by color or symbol) points in all plots. Typically, the marking variable will have two or possibly three categories; if more than six are used, then colors will begin to repeat. This item is used to turn marking off, to change the marking variable, or to add marks after a plot is drawn.

Selecting one of the first five of these items will give a dialog to choose the quantities to graph. These dialogs include new features. The user can select a variable in the "Mark by…" box. This variable will be used to define groups of points in the plot, and each group will be given a different color and symbol. You can switch to using colors or symbols by changing the `*mark-groups-with-color*` and `*mark-groups-with-symbol*` settings (see Section 4.1). Colors are better for the screen but symbols are needed with most printers; the default is to use both color and symbols. For example, if you "mark by" a variable called `sex`, with values 0 and 1, then all the points with `sex` = 0 will be marked in one color, and all the points with `sex` = 1 will be marked another color. If the marking variable has more than seven distinct values, you will get a dialog that asks if you really want to mark with it, and you will be given the opportunity to replace the marking variable by a discrete (i.e., grouped) version of it with the number of levels you specify. Once you choose a marking variable, it is used in all plots until you either use another one, you select the "Remove marks" menu item, or you set colors manually using the color palette in the top left corner of a plot. For other uses of marks, see Sections 7.1 and 7.7.

In the plot dialog you can also choose a variable to be used as "weights." If you fill in this area, then the weights will be used in fitting smoothers to the plot; see Section 7.1. Two push-buttons allow you to "jitter" the graph (which means to add a small amount of random error before graphing, so seeing over-plotted points is easier), and to remove the plot controls (you can get them back using a menu item from the plot's menu).

7.1 Two Dimensional Plots

The plot controls for 2D plots have been enhanced. A new plot control called "Options" allows you to change axis labels, tick marks and text, to frame the graph, and to add a curve to the graph using a self-explanatory dialog. You can

also add the transformation slide bars that usually appear only on scatterplot matrices using a check-box in the "Options" dialog. The slide bars that have pop-up menus are indicated with a triangle pointing down. In the parametric smoother slide bar (the one that initially has "OLS" above it), you can now fit using ordinary least squares (OLS), Huber M-estimation (with fixed tuning parameter), or using logistic regression, Poisson regression or Gamma regression. If you have specified weights, they are used in the fitting. For logistic regression, a plot of the proportion of successes on the vertical axis with weights equal to the number of trials will fit the correct logistic regression. If the number of trials equals one for all cases (for example, the response variable is a list of ones and zeroes representing success or failure), you do not need to specify weights. The "power curve" smoother has been moved from the nonparametric slide bar (the one that initially has "lowess" above it") to the parametric slide bar.

The smoothers available in the nonparametric smoothing slide bar include lowess, and a very primitive slice smoother, in which the range of x is divided into non-overlapping slices, and the mean is computed in each slice. This is exactly the smoother that is used by Sliced Inverse Regression (SIR)—see *IRG*, p. 119–133. Also available is a spline smoother; details on this will be given later. The kernel smoothers, which were rarely used in version 1, are no longer available; other smoothers will be added later. The "Mean+−SD" slider uses lowess to compute an estimate of the mean function, then computes the squared residuals from the lowess smooth, and smooths them to estimate the standard deviation. The lines shown on the plot are the estimated mean and the estimated mean plus and minus one standard deviation.

For each smoother, you can now select the "Fit by marks" option, if the points are marked by group. The smooth will then be computed and fit separately for each group.

An additional mouse mode called "Slicing" has been added. If you select this mode, you will have the same cursor you get for brushing, and indeed you can resize the brush in the usual way. The difference with brushing is that summary statistics for the points in the brush are printed on the plot, namely the number of points in the slice and their mean and standard deviation on the vertical axis. You can get these numbers printed in the text window, as well as the mean and standard deviation on the horizontal axis, by control-shift clicking the mouse.

7.2 Three Dimensional Plots

One plot control has been added to 3D plots that is a generalization of the parametric smoother slide bar for 2D plots. Use this slide bar to fit 3D surfaces. The choices are linear, full quadratic, or any other quadratic obtained by dropping terms from the full quadratic. An additional choice is to fit

$$\mathrm{E}(y|x,z) = \theta_0 + \theta_1 x + \theta_2 z + \theta_3(\theta_1 x + \theta_2 z)^2 \tag{1}$$

which is a one-dimensional quadratic model. This mean function is nonlinear, and so computing it takes much longer than the other functions. Parameter estimates for fitting mean function (1) can be obtained as described in Section 8.1.

7.3 Scatterplot Matrices

Each of the transformation slide bars now has a pop-up menu to add a power choice to the slide bar, to add the transformed values to the data set, or to select a transformation to make the variable as normal as possible (using a one-dimensional Box-Cox procedure). A new "Transformations" plot control includes a number of self-explanatory options. It also allows selecting a number of variables and transforming them simultaneously to achieve multivariate normality using a multivariate generalization of The Box-Cox procedure.

7.4 Boxplots

The dialog allows a choice of several variables and a conditioning variable. If you do not select a conditioning variable, you will get a parallel boxplot for the variables you have selected. If you do select a conditioning variable, you will get a boxplot of the first variable split according to levels of the conditioning variable. A plot control allows printing a corresponding analysis of variance table at the bottom of the plot. Weights, if set, are used in computing the analysis of variance.

7.5 Multi-panel Plots

This item allows examining a number of 2D plots with one axis common, and the other axis variable. For example, one might assign the response to the fixed vertical axis, and all the predictors to the variable horizontal axis to view partial response plots, or reverse the assignment with the response on the horizontal axis to view all inverse partial response plots. It would be nice to see these plots in

parallel in one window, but this is not supported in the current version of *Xlisp-Stat*.

7.6 Probability Plots

This is used to study the distribution of a random sample. The ordered data are plotted against quantiles from a hypothesized distribution (chosen from the normal, half-normal, chi-squared, or t distributions); if the sampling distribution of the data is the same as the hypothesized distribution, then the plot should be approximately straight.

7.7 Conditioning Plots

If you have a marking variable in a graph, you can click the mouse on one of the values of the marking variable, and only points in that group will remain visible. Click on the name of the marking variable to restore all the data. This is useful for examining conditional distributions.

8 Fitting Models

Unlike *R-code*, *Arc* does not fit models automatically. You fit a model by selecting one of the fit items from the Graph&Fit menu (*R-code* only fit linear least squares regressions). The choices are:

Fit linear LS... Used to fit linear least squares regression, or normal generalized linear models. The dialog can be used to choose a mean function, which is the inverse of the link function commonly discussed in generalized linear models. The identity link gives usual linear models, but the inverse and exponential mean functions, corresponding to the inverse and log links, are also available. You can also choose a "Full quadratic" to fit a full second-order model in the predictors or a "1D-quadratic" to fit a one-dimensional quadratic in the predictors, which is equivalent to (1) with $p = 2$ predictors.

Fit binomial response... Binomial generalized linear models. The response is the number of successes. You must also specify the number of trials and the mean function (with the logistic mean function the default).

Fit Poisson response... Poisson generalized linear models. You must specify the mean function (with the exponential mean function the default).

Fit gamma response... Gamma generalized linear models. You must specify the mean function (with the inverse mean function the default).

Fit nonlinear LS... Used to fit nonlinear least squares regression. This method requires specification of a mean function for the data, and you do this in a dialog box that will be provided. Type an expression like `th0*(1-th1*(exp(th2*x1)))`, where the parameters are given by `th0`, `th1`..., and the predictors are `x1`, `x2`,.... Any valid mathematical expression is acceptable. Initial estimates for the parameters are also needed. Derivatives are computed symbolically.

Inverse regression... Fit inverse regression (Quadratic fit, Cubic fit, Sliced Inverse Regression (SIR), pHd, or SAVE). More details on these will be given later.

Graphical regression... Details on this item are discussed in Chapter 20 of Cook and Weisberg (1999).

8.1 Fit Linear LS

The dialog for linear models is similar to *R-code*, except you are now permitted to choose a *kernel mean function*. The usual choice is the identity function, giving usual linear models. For those familiar with generalized linear models, the kernel mean function is the inverse of the usual link function, so, for example, the exponential kernel mean function is the same as the log link function. When you create a model, you will get a Model menu, mostly containing items from the *R-code* version 1 model menu.

8.2 Generalized Linear Models

The menu for generalized linear models is similar to the linear regression menu, except you must specify a kernel mean function and for binomial data you must also specify the number of trials.

8.3 Nonlinear Models

To specify a nonlinear model, you must specify both a mean function and starting values for the parameters (if you do not specify starting values, they are all set to zero). A mean function is a text string like `"th0+th1*x1+th2*(1+exp(th3*x2))"`, where the parameters are `th0`, `th1`, ..., and the variables are `x1`, `x2`, ... Other text strings that can be used for the parameters are `theta`, `gam`, `gamma`, `a`, `alpha`, `b`, and `beta`, while `z`'s can be used for the variables as well as `x`'s. You can specify a mean function in three ways:

1. After selecting "Fit Nonlinear LS...", type the mean function into the resulting mean function dialog. This is harder than it sounds, and errors are common. Alternatively, a slightly easier method is to type the command

   ```
   (def m "th0+th1*x1+th2*(1+exp(th3*x2))")
   ```

   and then to type `m` into the "Fit Nonlinear LS..." mean function dialog.

2. On the data file, add a construction like

   ```
   begin mean functions
   th0+th1*x1+th2*(1+exp(th3*x2))=(10 15 12)
   end mean functions
   ```

   The syntax is mean function = starting values. Alternatively, if you create a mean function using the dialog and then save the datafile, the mean function and starting values will be saved.

3. You can type in a command. If `d` is the name of the data set, then type

   ```
   (send d :mean-function
       "th0+th1*x1+th2*(1+exp(th3*x2))" (10 15 12))
   ```

The nonlinear regression menu is considerably different from the others. Here is a brief description of the items in the nonlinear model menu.

Display Fit Print the estimates, standard errors and a summary analysis of variance table on the screen.

Display Variances Print the matrix $\hat{\sigma}^2(V^TV)^{-1}$, where V is the matrix of first-derivatives of the mean function with respect to the parameters. This is the approximate variance-covariance matrix of the coefficient estimates. This matrix may not be meaningful if the distribution of the estimates is far from the normal distribution. Also printed by this command is this matrix converted to a correlation matrix.

Fit model to groups... This allows the data to be split into groups using one of the variates (sex, for example), and then the specified non-linear model is fit to both groups separately.

Subset model... Select this item to produce a complicated dialog to allow the user to build a sub-model of the nonlinear model by setting some of the parameters. A parameter can be set "free" (or left blank) to allow it to be estimated; it can be set "current" to fix it at its current value; it can be set to any numerical value; or two or more parameters can be set equal to one another. Starting values are determined from the current fit.

Model checking plots... Produce model checking plots (see Section 9).

Likelihood region... If the user selects one parameter in the resulting dialog, a *confidence curve* for that parameter is drawn. If two parameters are chosen, then a *contour plot* is drawn showing simultaneous confidence regions for the two parameters. Controls on the contour plot allow adding more contours and adding more points to the grid (which makes the contour plot smoother).

Wald region... Select one or two parameters to show the 95% confidence interval or region based on the normal approximation. For one parameter, the confidence curves are two straight lines, and for two parameters the confidence region is elliptical.

2D Likelihood-Wald Discrepancies This method compares the Wald and likelihood regions. For each pair of parameters, many points on the boundary of a 95% Wald region are found, and at each of these points the likelihood function is evaluated to find the "actual" level at that point. The reported values are the minimum and maximum coverage found on the boundary of the region. If the minimum and maximum are close to the nominal 95%, then the Wald approximation is close to the likelihood region.

Plot response function This item only works if there is just one predictor. It produces a scatterplot of the data, with the fitted mean function superimposed; this allows a graphical appraisal of the model.

Distances Plot Produce a plot of Cook's distance against case number. Cook's distance is computed based on the normal approximation at the estimates, so it is a one-step estimate. A pop-up menu gives a choice of other ways of computing an influence statistic, either by iterating a number of times to get closer to the exact Cook's distance, or by using a different metric in place of Cook's distance.

Parameter Plot Produce a parameter plot, a generalization of added-variable plots for nonlinear models.

ALP residual plots These are adjusted linear predictor residual plots and are the same as in linear models (see *IRG*, p. 179–182).

Nonconstant variance plot This is the same as in linear models (see *IRG*, p. 182–188).

Options... Set a number of numerical options to aid in obtaining convergence.

Curvatures Compute and print parameter effects and intrinsic curvatures. This computation can be very intensive, as it requires at least n^2 storage locations, and uses numerical second-derivatives.

8.4 Inverse Regression

Inverse regression models can also be fit from the data set menu. See *RG*, Chapter 8. The pHd method currently contains some experimental output not explained in the book. In addition, the inverse regression can be estimated by quadratic or cubic regression of the predictors on the response. All this methodology will be discussed more fully at a later date.

9 The Model Menu

All models have an associated menu with the same name as the model. For linear and generalized linear models, the model menus are similar to the model menu in *R-code*, but some of the items on the menu have changed. The "Display variances"

item prints $\hat{\sigma}^2(X^TWX)^{-1}$, where W is a diagonal matrix of prior weights for linear models, and computed weights for generalized linear models.

The "Examine submodels..." item can be used to get fit statistics, really just residual sums of squares or deviances, for a sequence of models. The user specifies a "base model", the set of predictors always included, and the remaining variables to add. Note that the order the predictors are listed matters. Computations can be done in four ways: forward selection, in which variables are added to the base model one at a time, each time improving the fit as much as possible; backward elimination, each time removing a variable that degrades the fit statistics as little as possible; sequential, the default, in which the variables are added in the order given; and delete one at a time, in which the likelihood ratio test for each variable fit last is computed.

The "model" menu contains two new graphical items, "Model checking plots," and "Uncorrelated scatterplot matrix." This latter item will be described at a later date.

The item "Model checking plots" is discussed in R. D. Cook and S. Weisberg (1997), "Graphics for assessing the adequacy of regression models," *Journal of the American Statistical Association*, 92, 490-499. Model checking plots always consist of points and two smooths.

The points are a scatterplot of the response variable on the vertical axis versus a linear combination of the predictors or other variables on the horizontal axis. Using a dialog, the user selects the variables that will be combined to produce the linear combinations on the horizontal axis. If all predictors are continuous, then the selected variables will generally be all the predictors; if some predictors are factors, then these may be excluded. The dialog also includes three radio buttons to choose how the plots are to be constructed. Selecting the default "Marginal plots" will produce plots of the response versus each of the predictors in turn; a slider on the plot allows moving between predictors. Selecting "pHd (on residuals) directions" produces a series of plots in which the plotting directions are all uncorrelated with each other and also uncorrelated with the response. The way the uncorrelated directions are chosen depends on the method pHd, described elsewhere. If the model fit is not adequate, the first pHd direction is likely to show a problem. The third choice is "random plots," in which random linear combinations of the selected predictors are used to generate the plots; the random direction can be changed by pressing the mouse button in the slide bar on the graph.

The two superimposed smooths consist of one that directly smooths the data and another that smooths the model (see the paper for details). The smooths are added to the plot using a slider at the bottom left of the plot; a lowess smooth

with common smoothing parameter is fit for each. If the model is adequate for the data, then the two smooths should agree for all possible choices of horizontal axis. Simple variance smooths can also be fit using the pop-up menu for the slide bar.

10 Miscellaneous

10.1 arcinit.lsp

This file must be present for *Arc* to run. Do NOT change this file!

10.2 Functions

Several functions have been added to *Arc* that can help in creating data. The function `gl` can be used to generate a variable of levels for data that are patterned; for example `(gl 2 3 12)` will generate a vector of two levels in blocks of three, and of length 12, giving `(0 0 0 1 1 1 0 0 0 1 1 1)`. The function `cut` will create a categorical variable from a continuous variable, and `recode`, `recode-value` and `recode-values` can be used to recode a variable. You can get more help using the "Help" item in the Arc menu.

10.3 Multiple Data Sets

As before, you can work with many data sets at the same time, but *only one data set and its models will be displayed in the menu-bar*. To switch between data sets, simply select the data set's name from the Arc menu. One of the effects of this change is that many data sets can have a variable with the same name without any conflicts.

10.4 A new :prediction method

In the *R-code*, the method `:lin-combination` was introduced to obtain the estimate of a linear combination of estimates and its standard error. This method could be used to get predictions as well. An additional method called `:prediction` is now included. If you have a linear model called `L1` with 3 predictors, the typed command

```
(send l1 :prediction 23 12 4 :weight 2.1)
```

will compute a prediction with the first predictor equal to 23, the second equal to 12 and the third equal to 4, assuming that the variance of the error for the new observation is $\sigma^2/2.1$. For unweighted problems, the :weight keyword can be ignored. There is now a menu item on the regression model menu that allows using this command without typing.

10.5 Comments

We would appreciate your sending email to arc@stat.umn.edu if you encounter any problems or have any suggestions.